UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/726,779	11/29/2000	Chris Cifra	5150-43700	1986

35690 7590 07/28/2006

MEYERTONS, HOOD, KIVLIN, KOWERT & GOETZEL, P.C.
700 LAVACA, SUITE 800
AUSTIN, TX 78701

EXAMINER	
PILLAI, NAMITHA	

ART UNIT	PAPER NUMBER
2173	

DATE MAILED: 07/28/2006

Please find below and/or attached an Office communication concerning this application or proceeding.

UNITED STATES PATENT AND TRADEMARK OFFICE

Commissioner for Patents
United States Patent and Trademark Office
P.O. Box 1450
Alexandria, VA 22313-1450
www.uspto.gov

BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

Application Number: 09/726,779
Filing Date: November 29, 2000
Appellant(s): CIFRA ET AL.

Jeffrey C. Hood
For Appellant

EXAMINER'S ANSWER

This is in response to the appeal brief filed 5/8/06 appealing from the Office action mailed 1/30/06.

(1) Real Party in Interest

A statement identifying by name the real party in interest is contained in the brief.

(2) Related Appeals and Interferences

The examiner is not aware of any related appeals, interferences, or judicial proceedings which will directly affect or be directly affected by or have a bearing on the Board's decision in the pending appeal.

(3) Status of Claims

The statement of the status of claims contained in the brief is correct.

(4) Status of Amendments After Final

The appellant's statement of the status of amendments after final rejection contained in the brief is correct.

(5) Summary of Claimed Subject Matter

The summary of claimed subject matter contained in the brief is correct.

(6) Grounds of Rejection to be Reviewed on Appeal

The appellant's statement of the grounds of rejection to be reviewed on appeal is correct.

(7) Claims Appendix

The copy of the appealed claims contained in the Appendix to the brief is correct.

(8) Evidence Relied Upon

6,298,474 B1	BLOWERS ET AL.	10-2001

(9) Grounds of Rejection

The following ground(s) of rejection are applicable to the appealed claims:

Claim Rejections - 35 USC § 102

The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> (e) the invention was described in (1) an application for patent, published under section 122(b), by another filed in the United States before the invention by the applicant for patent or (2) a patent granted on an application for patent by another filed in the United States before the invention by the applicant for patent, except that an international application filed under the treaty defined in section 351(a) shall have the effects for purposes of this subsection of an application filed in the United States only if the international application designated the United States and was published under Article 21(2) of such treaty in the English language.

Claims 1, 3-5, 7-14, 16-18, 20-27, 29-31, 33-40 and 42-59 are rejected under 35 U.S.C. 102(e) as being clearly anticipate by U. S. Patent No. 6, 298, 474 B1 (Blowers et al.).

Referring to claims 1 and 27, Blowers discloses a method for generating a computer program by receiving user input specifying a prototype, wherein the prototype comprises a series of functional operations, wherein at least one of the operations has an associated one or more parameters (column 1, lines 47-55). Blowers discloses that the input is to a prototyping application to which the user input specifies a prototype (column 3, lines 28-35). Blowers discloses functional operations representing the functions used for carrying out tasks with a first functional operation amongst a plurality of operations with associated parameters with the functional operation (column 3, lines 15-35). Each of the multiple nodes in a structure represents functions with parameters associated with these functions. Blowers discloses automatically generating the program that implements the prototype, in response to the specified prototype, wherein comprising automatically generating a graphical user interface for the program (column

3, lines 15-20). Blowers also discloses that automatically generating the graphical user interface comprises automatically creating one or more graphical user interface elements associated with the one or more parameters of the first functional operation (column 3, lines 26-33), wherein during execution of the program, at least one of one or more graphical user interface elements are displayed and are operable to receive user input or display the output (column 9, lines 7-11), wherein as seen in Figure 7, the user configures the parameter data wherein the execution of the program is evident in the display of the graphical representations of the program along with a results "blog" window both shown in Figure 7. Blowers discloses that the program is operable to execute independently of the prototyping application (column 3, lines 35-45), there being a distinctiveness between the program that carries out the task through the structure created in the hardware configuration and the machine vision system (column 4, lines 1-15). Blowers also teaches that the user is allowed to input during execution without interference from the prototyping application thereby showing independence. Blowers teaches that users can input by single step inputs during debugging when execution occurs along with user input, the user input independent of the prototyping application.

Referring to claims 3, 16, 29 and 42, Blowers discloses generating the program comprises automatically generating source code for the program without direct user input specifying the source code (column 2, lines 45-55).

Referring to claims 4, 17 and 30, Blowers discloses that the first functional operation has an associated input parameter (column 3, lines 28-34), wherein the

graphical user interface comprises a graphical user interface element for interactively providing program input specifying a value for the input parameter, as seen in Figure 7 (column 9, lines 7-9). Blowers discloses that the automatically created graphical user interface elements include a graphical user interface element for interactively receiving user input specifying a value for the input parameter (column 8, lines 61-67), where the task sequencer interface and Blowers further teaches that the user can input or configure parameters related to these nodes of the sequence.

Referring to claims 5, 18 and 31, Blowers discloses that the first functional operation has an associated output parameter, wherein the automatically created graphical user interface elements include a graphical user interface element for viewing program output indicating a value for the output parameter (column 3, lines 45-55).

Referring to claims 7, 20 and 33, as seen in by the code formed in text forms in within the graphical representation of Figure 6 of Blowers, the generated program is text-based. Blowers also teaches that it is well known to generate programs comprising source code for the program in a text-based programming language (column 1, lines 33-36).

Referring to claims 8, 21, 34 and 44, as seen in Figure 6, the automatically generated program is a graphical program, comprising a plurality of interconnected nodes that visually indicate functionality of the program. Blowers discloses automatically generating source code for the graphical program (column 3, lines 40-45).

Referring to claims 9, 22 and 35, Blowers discloses that the prototyping application interfaces with a programming environment application in order to perform the generation of the program (column 3, lines 40-44).

Referring to claims 10, 23 and 36, Blowers discloses determining the data type of the first parameter and automatically creating a first graphical user interface element associated with the first parameter and able to receive user input of the parameters, as seen in Figure 6, wherein the "Blob", "Acquire" are examples of controls wherein the control expresses an association based on the types of data that is being accessed.

Referring to claims 11 and 37, Blowers discloses the prototype specifying an image-processing algorithm, with the automatically generated program implementing this image-processing algorithm (column 2, lines 50-52).

Referring to claim 12, Blowers discloses automatically creating graphical user interface elements including a first graphical user interface element for receiving user input specifying a value for an input parameter of the first functional operation, wherein the value for the input parameter affects the image processing algorithm (column 3, lines 25-35).

Referring to claims 13, 26 and 39, Blowers discloses the graphical user interface including automatically created graphical user interface elements including a first graphical user interface element for viewing an output value for an output parameter of the first functional operation where the output value is determined by the image processing algorithm, as seen by the image shown in Figure 7 (column 3, lines 45-55).

Referring to claim 14, Blowers discloses a method for generating a computer program by receiving user input specifying a prototype through a prototyping environment application, wherein the prototype comprises a series of functional operations, wherein at least one of the operations has an associated one or more parameters, as seen in specifying a prototype in Figure 7 (column 2, lines 47-55 and column 9, lines 7-9). Blowers discloses the prototyping environment operable, the association caused by the specifying of the prototype by the user in the prototyping environment, wherein these parameters would be used to automatically generating the program that implements the prototype, in response to the specified prototype (column 3, lines 40-44), wherein comprising automatically generating a graphical user interface for the program (column 3, lines 15-20). Blowers also discloses that the graphical user interface comprises creating user interface controls associated with the one or more parameters (column 3, lines 1-5 and 24-30). Blowers also discloses that automatically generating the graphical user interface comprises automatically creating one or more graphical user interface elements associated with the one or more parameters (column 3, lines 26-33), wherein during execution of the program, at least one of the one or more graphical user interface elements is displayed and is operable to receive user input or display the output (column 9, lines 7-11), wherein as seen in Figure 7, the user configures the parameter data wherein the execution of the program is evident in the display of the graphical representations of the program along with a results "blog" window both shown in Figure 7. Blowers discloses that the program is operable to execute independently of the prototyping application (column 3, lines 35-45), there

being a distinctiveness between the program that carries out the task through the structure created in the hardware configuration and the machine vision system (column 4, lines 1-15). Blowers also teaches that the user is allowed to input during execution without interference from the prototyping application thereby showing independence. Blowers teaches that users can input by single step inputs during debugging when execution occurs along with user input, the user input independent of the prototyping application.

Referring to claim 24, Blowers discloses that the prototyping environment application is an image processing prototype environment application, as is seen in Figure 7 (column 11, lines 22-27). Blowers discloses the prototype specifying an image-processing algorithm, with the automatically generated program implementing this image-processing algorithm (column 2, lines 50-52).

Referring to claims 25 and 38, Blowers discloses that the automatically generated graphical user interface elements include elements that can receive input parameter values affecting the image-processing algorithm (column 4, lines 45-67).

Referring to claim 39, Blowers discloses that the automatically created graphical user interface elements include elements for viewing output parameters values determined by the image processing algorithm (column 9 lines 24-26).

Referring to claim 40, Blowers discloses a method for automatically generating a computer program by receiving a program information to a first application the program information specifying functionality of the computer program (column 1, lines 47-55). Blowers discloses automatically generating the program that implements the specified

functionality, in response to the program information, wherein comprising automatically generating a graphical user interface for the program (column 3, lines 15-20). Blowers also discloses that the graphical user interface comprises creating user interface controls for providing input to and/or viewing output from the program (column 3, lines 1-5 and 24-30). Blowers also discloses that automatically generating the graphical user interface comprises automatically creating graphical user interface elements associated with the one or more parameters (column 3, lines 26-33), wherein during execution of the program, the one or more graphical user interface elements are displayed and at least one of the one or more graphical user interface elements is operable to receive user input (column 9, lines 7-11), wherein as seen in Figure 7, the user configures the parameter data wherein the execution of the program is evident in the display of the graphical representations of the program along with a results "blog" window both shown in Figure 7, wherein this user configuration of the displayed parameters would indicate receiving user input. Blowers discloses that the program is operable to execute independently of the prototyping application (column 3, lines 35-45), there being a distinctiveness between the program that carries out the task through the structure created in the hardware configuration and the machine vision system (column 4, lines 1-15). Blowers also teaches that the user is allowed to input during execution without interference from the prototyping application thereby showing independence. Blowers teaches that users can input by single step inputs during debugging when execution occurs along with user input, the user input independent of the prototyping application.

Referring to claim 43, Blowers discloses as seen by the controls "Blob", "Acquire" on Figure 6, wherein the graphical user interface controls of the program corresponds to the parameters specified by the program information.

Referring to claim 45, Blowers discloses, as seen in Figure 6, the received program specifying a prototype, and the "if-then-else" statement specifically represent a test executive sequence and a state diagram, wherein the sequence is based on the current state of a distinct variable, as is a "if-then-else" statement is used for.

Referring to claim 46, Blowers discloses automatically generating a block diagram, wherein the block diagram comprises a plurality of interconnected nodes that visually indicate the functionality of the program as shown in Figure 6.

Referring to claim 47, Blowers discloses automatically generating a user interface panel, wherein the user interface panel comprises the graphical user interface elements (Figure 7).

Referring to claim 48, Blowers discloses receiving user input specifying a prototype, wherein the prototype comprises a series of functional operations, wherein at least one of the operations has an associated one or more parameters (column 1, lines 47-55). Blowers discloses that in response to receiving user input specifying the prototype, automatically generating a graphical program, wherein automatically generating the graphical program comprises a plurality of interconnected nodes that visually indicate functionality of the graphical program (column 8, lines 61-67). The plurality of interconnected nodes is operable to perform the series of functional operations (column 8, lines 61-67). Blowers discloses automatically generating a

graphical user interface for the graphical program, wherein the graphical user interface for the graphical program comprises at leas t one graphical user interface element which is associated with at least one of the one or more parameters, wherein the graphical program is interpretable or compilable (column 9, lines 7-11).

Referring to claim 49, Blowers discloses receiving user input specifying the prototype is performed by a development environment, the graphical program comprises generating second program instructions, wherein execution of the second program instructions is independent of execution of the development environment (column 9, lines 1-25 and column 7, lines 35-39).

Referring to claim 50, Blowers discloses receiving user input specifying a prototype, wherein the prototype comprises a series of functional operations, wherein at least one of the operations has an associated one or more parameters (column 1, lines 47-55). Blowers discloses that in response to receiving user input specifying the prototype, automatically generating a graphical program, wherein automatically generating the graphical program comprises automatically generating a plurality of interconnected nodes that visually indicate functionality of the graphical program (column 8, lines 61-67). Blowers discloses associating at least one of the one or more parameters with an element of the graphical user interface (column 9, lines 7-9).

Referring to claim 51, Blowers discloses receiving user input indicating the at least one of the one or more parameters, wherein associating at least one of the one or more parameters with the element of the graphical user interface is based on receiving

user input indicating the at least one of the one or more parameters (column 3, lines 25-35).

Referring to claim 52, Blowers discloses receiving user input specifying the prototype is performed by first program instructions, the graphical program generating second program instructions, wherein execution of the second program instructions is independent of execution of the first program instructions (column 9, lines 1-25 and column 7, lines 35-39).

Referring to claim 53, Blowers discloses displaying a prototyping environment user interface on a display of a computer system, wherein the prototyping environment user interface is usable to create a prototype (Figure 7). Blowers discloses receiving user input specifying the prototype, wherein the prototype comprises a series of functional operations, wherein at least one of the operations has an associated one or more parameters (column 3, lines 25-35). Blowers discloses automatically generating a program that implements the prototype, in response to the specified prototype, automatically generating the program comprises automatically generating a graphical user interface for the program, the graphical user interface of the program comprises at least one graphical user interface element which is associated with at least one of the associated one or more parameters, wherein the at least one graphical user interface element performs at least one of receiving information to the program and outputting information from the program during execution of the program (column 8, lines 60-67 and column 9, lines 1-25), wherein the graphical user interface of the program is independent of the prototyping environment user interface (column 7, lines 35-39).

Referring to claim 54, Blowers discloses that the program is interpretable or compilable (column 9, lines 11-12).

Referring to claim 55, Blowers discloses receiving user input to a development environment, wherein the user input specifies a series of functional operations, wherein at least one of the operations has an associated one or more parameters (column 8, lines 60-67). Blowers discloses automatically generating a program that implements the series of functional operations, in response to the user input, wherein the program execution of the program is independent of execution of the development environment (column 9, lines 1-25 and column 7, lines 35-39). Blowers discloses automatically generating the program comprises automatically generating a graphical user interface for the program, with at least one graphical user interface element which is associated with at least one of the associated one or more parameters, wherein the at least one graphical user interface element performs one or more of receiving information through the graphical user interface and outputting information from through the graphical user interface during execution of the program, wherein the program is interpretable or compilable (column 9, lines 1-25).

Referring to claim 56, Blowers discloses automatically generating the program comprises automatically generating source code to perform the functional operations of the prototype (column 4, lines 15-20). Blowers discloses that the source code is generated in a format that allows a user to edit the automatically generated source code (column 4, lines 64-67).

Referring to claim 57, Blowers discloses that functional operations also include a plurality of operations including a second functional operation with associated parameters (column 8, lines 61-67). Blowers discloses automatically generating graphical user interface elements associated with the parameters of the second functional operation (column 8, lines 61-67).

Referring to claim 58, Blowers discloses executing the automatically generated program including displaying the automatically generated graphical user interface independently of the prototyping application (column 3, lines 35-45), there being a distinctiveness between the program that carries out the task through the structure created in the hardware configuration and the machine vision system (column 4, lines 1-15).

Referring to claim 59, Blowers discloses automatically generating the program is operable to perform the functional operations of the prototype (column 8, lines 64-67).

Claim Rejections - 35 USC § 103

The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

Claims 6, 19 and 32 are rejected under 35 U.S.C. 103(a) as being unpatentable over Blowers.

Referring to claims 6, 19 and 32, Blowers discloses that a plurality of parameters

are associated with the first functional operation, wherein receiving user input specifying which of the plurality of parameters are desired to have associated graphical user interface elements (column 3, lines 28-31). Blowers discloses that the input is to a prototyping application to which the user input specifies a prototype (column 3, lines 28-35). Blowers also discloses that automatically generating graphical user interface comprises automatically creating graphical user interface elements associated with each specified parameter of the first functional operation (column 3, lines 31-35). Blowers discloses that the automatically created graphical user interface elements include a graphical user interface element for interactively receiving user input specifying a value for the input parameter (column 8, lines 61-67), where the task sequencer interface and Blowers further teaches that the user can input or configure parameters related to these nodes of the sequence. Blowers does not explicitly disclose not creating graphical user interface elements associated with unspecified parameters of the first functional operation but Blowers only discusses displaying the desired graphical user interface elements, which represent the parameters. It would have been obvious that the undesired and those not chosen by Blowers would not have graphical user interface elements associated with any parameters, for reasons that they need not be displayed. Blowers may not explicitly disclose that unnecessary graphical user interface elements will not be generated with the unspecified parameters. But based on the fact, that an undesired parameter would not be useful to the user and need not be displayed to the user, makes it obvious that such unnecessary components would not be created. Hence, it would have been obvious to one skilled in the art, at the

time of the invention to not create graphical user interface elements associated with unspecified parameters, which need not be displayed.

(10) Response to Argument

Claims 1, 11, 14, 24, 27, 37, 43, 47, 53, 54, 55, 57, 58, 59

The prototyping application has been referred to as the machine vision system of Blowers. Blowers teaches that the machine vision system and the components used for developing the program are separate as is referred to in Examiner's arguments (Final Office Action mailed on 1/30/06) and Appellant's arguments in the Appeal Brief (page 11, lines 1-4). The environment in which the program is developed includes the task sequence engine coupled with the runtime interface both responsible for creating and executing the program (Figure 3 and column 9, lines 13-18). The machine vision system in Figure 2 is a separate system, which is associated with the program, created but is separate from the program and is independently executed from the machine vision system (column 8, lines 28-34). The components including the task sequence engine and runtime interface engine are responsible for creating and executing the program, where these components are part of the development system (Figure 3), which is clearly separate from the machine vision system (Figure 2). Therefore, the program executed by the runtime interface engine is executed independently of the prototyping application, which is the machine vision system. See column 9, lines 13-25.

Blowers teaches dragging and dropping of icons, which is the user's input step for choosing the series of functional operations. The program that is generated as a result of choosing these functional operations includes automatically generating a

program that would alleviate the user being responsible for writing the program. Therefore, it is clear that Blowers teaches the automatic generation of a program. The user chooses the functional operations represented as icons that are to be part of the program but the code and detailed elements associated with these functional operations are automatically generated for execution so that the user is not held responsible for writing the code. Appellant argues that Blowers does not teach the tree structure is automatically generated. The claims do not recite that a tree structure is automatically generated. Blowers also discloses that the functional operations or tasks chosen are linked to generate a structure that includes the functionality with the graphical structure, which is to represent the program. The task sequence engine is responsible to link the functionality with the structure to generate a program, and create a link between the functionality and the nodes of the structure. The user may choose the functions but the task sequence engine is responsible for linking these functions that are separately chosen and creating a linked structure, which represents an entire program including the functions and parameters chosen by the user. See column 8, lines 60-67. The claims disclose that a graphical user interface associated with the program is automatically generated, which can also be interpreted as the results user interface which is automatically generated during execution of the associated program (Figure 9 and column 9, lines 16-20). Furthermore, Appellant argues that the tree structure is not independent of the development engine, which is not taught in the claims. The tree structure is created and executed using the task sequence engine of Figure 3, which allows for the program to be executed independently from the machine vision system of

Figure 2. Blowers discloses that the program code is automatically generated so as to alleviate the user from writing the code, with the creation of a sequence of commands which is generated representing the tree structure (column 11, lines 55-57). A sequence of commands is interpreted as code.

Claims 3, 16, 29 and 42

Blowers has clearly taught that the purpose of the system is to ensure that the user does not write the code but allows for options to be chosen to generate a program automatically. The generated program contains a sequence of commands that represents code. See column 2, lines 47-55 and column 11, lines 55-57. The user input involves selection of functional operations that represent a prototype with these selections leading to the generation of a program. The selections are made by the user to generate a program without the user being required to write the code, the code being generated automatically. Blowers teaches that user input is used for choosing prototypes which combined with first program generates a new program which is carried out automatically. Blowers discloses that a program is generated automatically, where as is disclosed in the Appellant's arguments, the user provides the certain functional operations which is used along with program data to generate an program automatically. The program that is generated is done so without the user writing the code with Blowers teaching that the user involvement is with providing functional operations, which is then used with further program data to automatically generate a new program. The code or sequence of commands is automatically generated for execution. The program of Blowers is contained in a sequence file, with this file being

able to run and therefore created into a computer executable form. The program that is generated, with the sequence file has the ability to be executed. Source code is known to be a collection of files that can be converted from a file format into an equivalent computer executable form, which has been done so for the Blower's sequence file. See column 9, lines 14-20. The ability to execute the data discloses that Blowers does teach the generation of the source code which is generated automatically with the combination of user input and further programming data.

Claims 4, 5, 7, 12, 17, 20, 25, 30, 31, 33 and 38

Each of the parameters described in Figure 4 have associated graphical user interface elements, through which the user can input to specify the parameters. Figure 5 (reference numbers 62, 68, 64) teach examples of further graphical user interface elements that the user can access for input of further parameters of the program that is automatically generated (column 9, lines 1-6). Blowers discloses the use of the tools and icons, which are graphical user interface elements that can be accessed by the user to input parameters associated with the program. Blowers discloses that the program is executed and parameters related to this program are then automatically generated and displayed to the user for input. Blowers has indicated that the generation of the parameters is carried out automatically when the program is executed. These parameters are displayed to the user for input, therefore the parameters are graphical user interface elements that are displayed for receiving user input. The program contains functional operations for defining a prototype with parameters that further define this prototype, which is generated when the program is executed. See

column 10, lines 33-35. The program independently operating from a prototyping application has been previously discussed. Blowers discloses that Figure 6 contains the program data including text-based information that represents the program. The program is automatically generated as previously discussed with the data being text-based programming data that includes text information that can be executed as a program.

Claims 8, 21, 34, 44, 46

Blowers does teach automatically generating a graphical program, with Blowers allowing the users to input various operations and parameters and the task sequencer engine being responsible for linking and generating the tree structure. The task sequence engine is responsible for automatically generating the graphical program. See column 8, lines 65-67. Blowers discloses that the generated sequence of source code, the source code being data in a file that can be compiled to become a computer executable file, where Blowers teaches the execution of this sequence of commands. See column 11, lines 55-67. The sequence of commands stored as the tree structure is automatically created by the task sequence engine. Blowers may disclose that the user has means for choosing functional operations and further dragging and dropping of these operations represented as icons but Blowers also teaches that all functional tasks are linked to a tree structure, representing the graphical program by the task sequence engine which is how the graphical program is generated. The user of Blower may choose the prototype, including functions and machine vision tasks desired to be

included in the program, but the task sequence engine carries out the generation of and linking of the program to the tree structure. See column 8, lines 65-67.

Claims 9, 22, 35

Blowers discloses that when the program is executed, parameters that are associated with the program are automatically generated on a display for the user to input to (column 10, lines 33-35). A parameters section is viewable to the user, where this section is automatically generated when the program is executed. The parameters being viewable will include graphical user interface elements that can receive user input. Each of the parameters have a distinct type, the parameters discussed in this section including input channel, input value and time out parameters, each of them representing various data types and any of them representing a first data type for parameters.

Claims 10, 23, 36

The prototyping application being the machine vision system interfaces with the programming environment, which is used to develop and execute the program to carry out functions that are related to the machine vision tasks. Blowers discuss how the machine vision system and the programming components used to create the program work together to carry out the tasks required for this system. See column 8, lines 28-33.

Claims 13, 18, 26, 39

Blowers discloses results windows which displays graphical user interface elements that represent output values for output parameters of the functional

operations, the output being displayed when the program has been executed. This output being displayed with the execution of the program shows an association with the program, the output value also being determined by the image-processing algorithm. The results windows related to the blob and further vision tools and the description including further image processing algorithm functions (column 9, lines 2-5) disclose that the output value is determined by the image processing algorithm. Image processing algorithms include functions that represent image processing functionalities including process image, acquiring image from a camera, blobs, all being examples of image processing algorithms (column 9, lines 2-5). The results generated as the output that displayed with graphical user interface elements are related to functional operations that are related to image processing algorithms including blobs, vision tools and image processing. See column 9, lines 1-15.

Claim 40

Blowers discloses a machine vision system and an application software that is created that can be used with a machine vision system. Blowers discloses two distinct components describing a first machine vision system in Figure 2, which represents the first application and components used for creating the program the components defined in Figure 3. The task sequence engine creates and executes the program independently of the machine vision system in Figure 2.

Claim 45

Blowers clearly does teach that the received program information specifies a prototype (column 3, lines 19-21), the prototype related to machine vision tasks. Blowers discloses the tree structure being created to test data and also serving as a state diagram to describe a state of the functions that are associated with the program. Blowers specifically discloses that the task sequence engine configures and tests the flow of the program, thereby teaching the test executive sequence. The test executive sequence being the program, which tests the sequence and flow of the components in the program. See column 8, lines 60-62. The program information also specifies a state diagram, where the "if-then-else" statement including in Figure 6 of the program describes a state diagram, teaching iteration through different states based on parameters. The "if-then-else" statement determines the next step that is to be taken based on current conditions of a parameter where if a certain condition were occurring currently, the next event would be a first event, otherwise the next event would be a different second event. This teaches how the "if-then-else" teaches a state diagram component.

Claims 48, 50, 51

As previously disclosed, Blowers does disclose automatically generating the program. In order for a program to be created, the user may write the program manually or a system may generate the program automatically with minimal user input to choose certain parameters or operations that may be associated with the user. Blowers has clearly avoided the user manually inputting the program and has taught

that the user's role is to choose certain functional operations and parameters associated with certain functions to generate a program automatically. Blowers discloses that the task sequencer engine is responsible for linking functional operations to a tree structure, thereby generating the program that is represented as the tree structure (column 8, lines 65-67). Blowers also describes that these tree structures include nodes (column 3, lines 18-19), with nodes having a connection, the tree structure connecting all the nodes to visually indicate the program. The plurality of interconnected nodes is operable to perform the series of functional operations. See column 3, lines 30-35.

Claims 49, 52

The task sequencer engine is responsible for generating and executing the program as is previously disclosed with the development environment being the machine vision tasks which receives inputs and from which the task sequence independently executes the program from the development environments.

Claim 56

Blowers discloses teaching allowing the user to change parameters which would be editing the automatically generated source code, with Blowers further teaching additional editing steps for making changes to the automatically generated source code. See column 12, lines 44-45.

Claims 6, 19, and 32

Each of the parameters described in Figure 4 have associated graphical user interface elements, through which the user can input to specify the parameters. Figure

5 (reference numbers 62, 68, 64) teach examples of further graphical user interface elements that the user can access for input of further parameters of the program that is automatically generated (column 9, lines 1-6). Blowers discloses the use of the tools and icons, which are graphical user interface elements that can be accessed by the user to input parameters associated with the program.

(11) Related Proceeding(s) Appendix

No decision rendered by a court or the Board is identified by the examiner in the Related Appeals and Interferences section of this examiner's answer.

For the above reasons, it is believed that the rejections should be sustained.

Respectfully submitted,

Namitha Pillai
Assistant Examiner
Art Unit 2173
July 21, 2006

Conferees:

Kristine Kincaid
Supervisory Patent Examiner
Art Unit 2174
July 21, 2006

Weilun Lo
Supervisory Patent Examiner
Art Unit 2179
July 21, 2006